INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **61**:1977–1991
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.1068

A general advancing front technique for filling space with arbitrary objects

Rainald Löhner[1,*,†] and Eugenio Oñate[2]

[1]*School of Computational Sciences, MS 4C7 George Mason University, Fairfax, VA 22030-4444, U.S.A.*
[2]*CIMNE, Universidad Politécnica de Catalunya Barcelona, Spain*

SUMMARY

An advancing front space-filling technique for arbitrary objects has been developed. The input required consists of the specification of the desired mean point distance in space and an initial triangulation of the surface. One object at a time is removed from the active front, and, if possible, surrounded by admissible new objects. This operation is repeated until no active objects are left. Two techniques to obtain maximum packing are discussed: closest object placement (during generation) and move/enlarge (after generation). Different deposition or layering patterns can be achieved by selecting the order in which objects are eliminated from the active front. Timings show that for simple objects like spheres the scheme is considerably faster than volume mesh generators based on the advancing front technique, making it possible to generate large ($> 10^6$) yet optimal clouds of points in a matter of minutes on a PC. For more general objects, the performance may degrade depending on the complexity of the penetration checks. Several examples are included that demonstrate the capabilities of the technique.

KEY WORDS: grid generation; mesh free techniques; discrete element method; SPH; finite point method

1. INTRODUCTION

Many simulation techniques in computational mechanics require a space-filling cloud of arbitrary objects. For the case of 'gridless' or 'mesh free' partial differential equation (PDE) solvers (see References [1–9]) these are simply points. For discrete element methods (see References [10–13]), these could be spheres, ellipsoids, polyhedra, or any other arbitrary shape. The task is therefore to fill a prescribed volume with these objects so that they are close but do not overlap in an automatic way.

*Correspondence to: Rainald Löhner, School of Computational Sciences, MS 4C7 George Mason University, Fairfax, VA 22030-4444, U.S.A.
†E-mail: rlohner@gmu.edu

Received 22 August 2003
Revised 29 December 2003
Accepted 28 January 2004

Several techniques have been used to place these objects in space. The so-called 'fill and expand' or 'popcorn' technique [13] starts by first generating a coarse mesh for the volume to be filled. This subdivision of the volume into large, simple polyhedra (elements), is, in most cases, performed with hexahedra. The objects required (points, spheres, ellipsoids, polyhedra, etc.) are then placed randomly in each of these elements. These are then expanded in size until contact occurs or the desired fill-ratio has been achieved. An obvious drawback of this technique is the requirement of a mesh generator to initiate the process. A second class of techniques are the 'advancing front' or 'depositional' methods [14, 15]. Starting from the surface, objects are added where empty space still exists. In contrast to the 'fill and expand' procedures, the objects are packed as close as required during introduction. Depending on how the objects are introduced, one can mimic gravitational or magnetic deposition, layer growing, or size-based growth. Furthermore, so-called radius growing can be achieved by first generating a coarse cloud of objects, and then growing more objects around each of these. In this way, one can simulate granules or stone.

Here, we present a scheme that allows for the direct generation of clouds of arbitrary objects with the same degree of flexibility as advanced unstructured mesh generators [16–26]. The mean distance between objects (or, equivalently, the material density) is specified by means of background grids, sources and density attached to CAD-entities. In order not to generate objects outside the computational domain, we assume an initial triangulation of the surface that is compatible with the desired mean distance between objects specified by the user. Starting from this initial 'front' of objects, new objects are added, until no further objects can be introduced. Whereas the advancing front technique for the generation of volume grids removes one face at a time to generate elements, the present scheme removes one object at a time, attempting to introduce as many objects as possible in its immediate neighbourhood.

2. THE ALGORITHM

Assume as given

- A specification of the desired mean distance between objects in space, as well as the mean size of these objects. This is done here through a combination of background grids, sources and mean distance to neighbours attached to CAD-data (see References [14, 25] for more details).
- An initial triangulation of the surface, with the face normals pointing towards the interior of the domain to be filled with points.

With reference to Figure 1, which shows the filling of a simple 2-D domain with ellipsoids, the complete advancing front space-filling algorithm may be summarized as follows:

- Determine the required mean point distance for the points of the triangulation;
- `while`: there are active objects in the front:
  - Remove the object `ioout` with the smallest specified mean distance to neighbours from the front;
  - With the specified mean object distance: determine the coordinates of `nposs` possible new neighbours. This is done using a stencil, some of which are shown in Figure 2;
  - Find all existing objects in the neighbourhood of `ioout`;

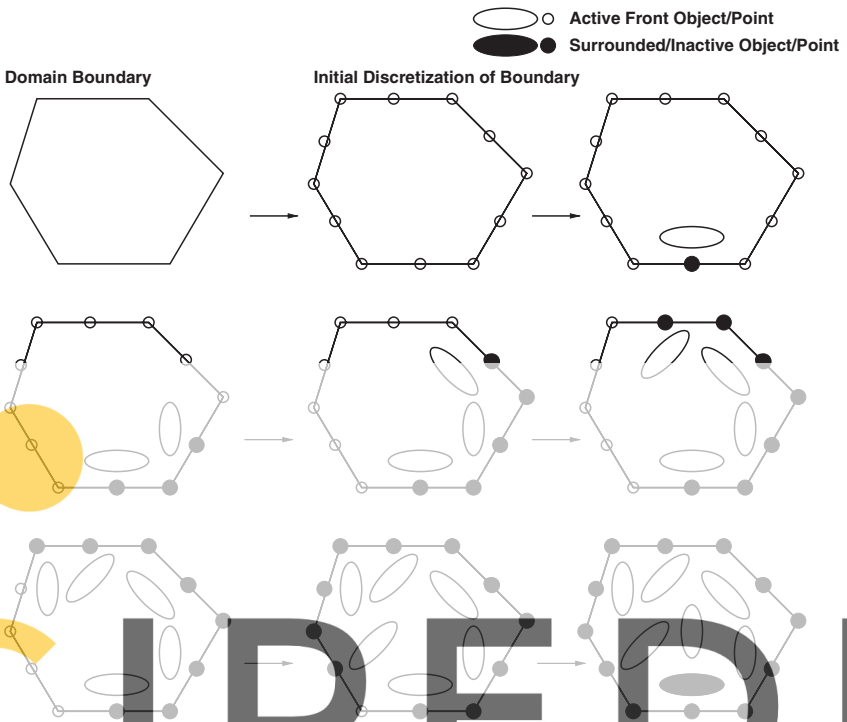

Figure 1. Advancing front space-filling with ellipses.

- o Find all boundary faces in the neighbourhood of `ioout`;
- o do: For each one of the possible new neighbour objects `ionew`:
  - If there exists an object closer than a minimum distance `dminp` from `ionew`, or if the objects are penetrating each other:
    ⇒ skip `ionew`;
  - If the object `ionew` crosses existing faces:
    ⇒ skip `ionew`;
  - If the line connecting `ioout` and `ionew` crosses existing faces:
    ⇒ skip `ionew`;
  - Determine the required mean point distance for `ionew`;
  - Increment the number of objects by one;
  - Introduce `ionew` to the list of coordinates and store its attributes;
  - Introduce `ionew` to the list of active front objects;
- o `enddo`
- `endwhile`

The main search operations required are

- Finding the active object with the smallest mean distance to neighbours;
- Finding the existing objects in the neighbourhood of `ioout`;
- Finding the boundary faces in the neighbourhood of `ioout`.

These three search operations are performed efficiently using heap-lists, octrees and linked lists respectively (see References [16–26] for more details).

3. POINT STENCILS

A number of different stencils may be contemplated. Each one of these corresponds to a particular space-filling point/object configuration. The simplest possible stencil is the one that only considers the 6 nearest neighbours on a Cartesian grid (see Figure 2(a)). It is easy to see that this stencil, when applied recursively with the present advancing front algorithm, will fill a 3-D volume completely. Other Cartesian stencils, which include nearest neighbours with distances $\sqrt{2}$ and $\sqrt{3}$ from `ioout` are shown in Figures 2(b), (c). The 'tetrahedral' stencil shown in Figure 2(d) represents another possibility. Furthermore, one can contemplate the use of random stencils, i.e. the use of n randomly selected directions to place new objects close to an existing one. For the generation of points and spheres, it was found that the 8-point stencil leads to the smallest amount of rejections and unnecessary testing.

In many instances, it is advisable to generate 'body conforming' clouds of points in the vicinity of surfaces. In particular, finite point [6, 7, 9] or smooth particle hydrodynamics (SPH) [5] techniques may require these 'boundary layer point distributions'. Such point distributions can be achieved by evaluating the average point-normals for the initial triangulation. When creating new points, the stencil used is rotated in the direction of the normal. The newly created points inherit the normal from the point `ioout` they originated from.

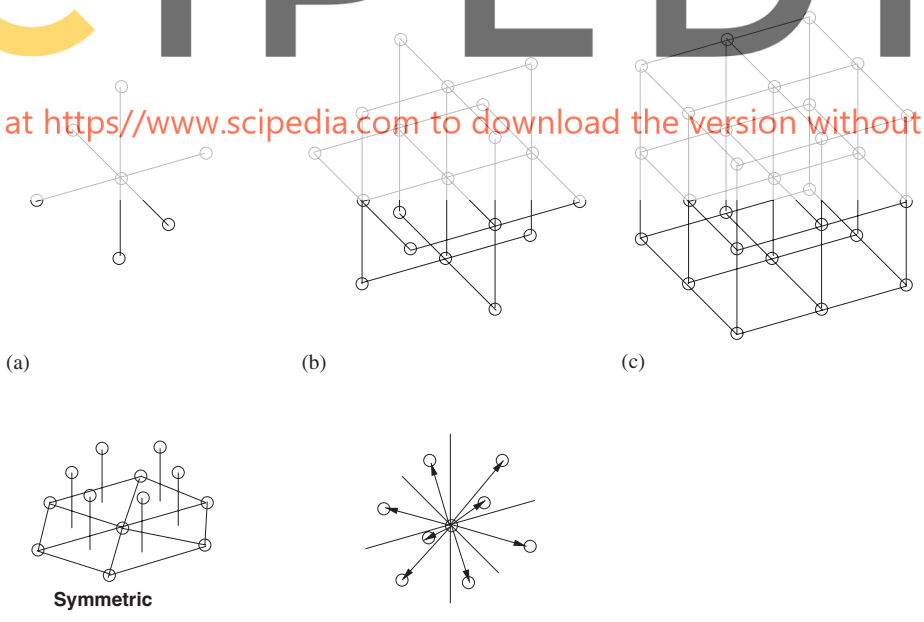

Figure 2. Point stencils: (a) Cartesian [4]; (b) Cartesian [9]; (c) Cartesian [23]; (d) tetrahedral [27]; and (e) random.

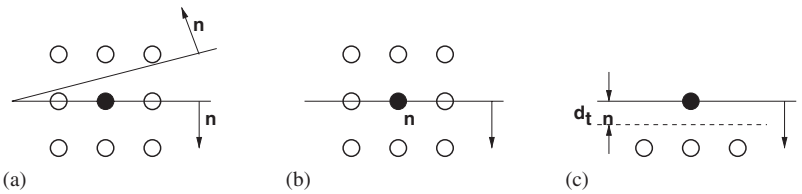

Figure 3. Boundary consistnecy checks: (a) obtain stencil for ioout and close faces; (b) filter faces pointing away from ioout; and (c) retain points inside computational domain. ●, ioout; ○, ionew.

4. BOUNDARY CONSISTENCY CHECKS

A crucial requirement for a general space-filling object generator is the ability to only generate objects in the computational domain desired. If we assume that the object to be removed from the list of active objects `ionew` is inside the domain, a new object `ionew` will cross the boundary triangulation if it lies on the other side of the plane formed by any of the faces that are in the proximity of `ionew` and can see `ionew`. This criterion is made more stringent by introducing a tolerated closeness or tolerated distance d_t of new objects to the exterior faces of the domain. Testing for boundary consistency is then carried out using the following algorithm (see Figure 3):

- Obtain all the faces close to `ioout`;
- Filter, from this list, the faces that are pointing away from `ioout`;
- `do`: For each of the close faces:
  - Obtain the normal distance d_n from `ionew` to this face;
    `if`: $d_n < 0$: `ionew` lies outside the domain
    ⇒ reject `ionew` and exit;
    `elseif`: $d_n > d_t$: `ionew` if far enough from the faces
    ⇒ proceed to the next close face;
    `elseif`: $0 \leqslant d_n \leqslant d_t$: obtain the closest distance d_{min} of `ionew` to this face;
    `if`: $d_{min} < d_t$: `ionew` is too close to the boundary
    ⇒ reject `ionew` and exit;
    `endif`
- `enddo`

Typical values for d_t are $0.707 d_0 \leqslant d_t \leqslant 0.9 d_0$, where d_0 denotes the desired mean average distance between points.

5. MAXIMUM COMPACTION TECHNIQUES

For SPH and finite point applications, the use of a stencil is sufficient to ensure a proper space filling (discretization) of the computational domain. However, many applications that consider not points but volume-occupying objects such as spheres, ellipsoids and polyhedra, require a preset volume fraction occupied by these objects, and, if possible, a minimum number of contacts. The modelling of discontinua via Discrete Element Methods represents a

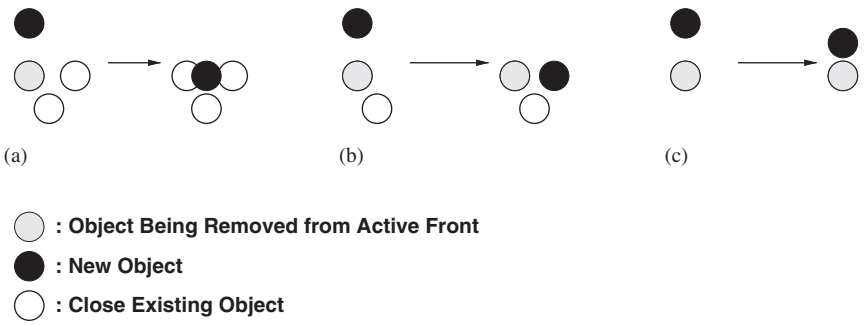

Figure 4. Closest object placement: (a) move to 3 neighbours; (b) move to 2 Neighbours; and (c) move to 1 Neighbour. ○, object being removed from active front; ●, new object; ○, close existing object.

typical class of such applications. Experience indicates that the use of point stencils does not yield the desired volume fractions and contact neighbours. Two complementary techniques have proven useful to achieve these goals: closest object placement and move/enlarge post-processing.

5.1. *Closest object placement*

Closest object placement attempts to position new objects as close as possible to existing ones (see Figure 4). The initial position for new objects is taken from a stencil as before. The closest three objects to the new object position and the object being removed from the active front are then obtained. This does not represent any substantial increase in effort, as the existing objects in the vicinity are anyhow required for closeness/penetration testing. Starting from the three closest objects, an attempt is made to place the new object in such a way that it contacts all of them, does not penetrate any other objects and resides inside the computational domain. If this is not possible, an attempt is made with the two closest objects. Should this also prove impossible, an attempt is made to move the new objects towards the object being removed from the active front. If this attempt is also unsuccessful, the usual stencil position is chosen.

5.2. *Move/enlarge post-processing*

Whereas closest object placement is performed while space is being filled with objects, post-processing attempts to enlarge and/or move the objects in such a way that a higher volume ratio of objects is obtained, and more contacts with nearest neighbours are established. The procedure, shown schematically in Figure 5, may be summarized as follows:

- `while`: objects can be moved/enlarged:
- `do`: loop over the objects `ioout`:
  - Find the closest existing objects of `ioout`;
  - Find all boundary faces in the neighborhood of `ioout`;
  - Move the object away from the closest existing objects so that:
    - The minimum distance to the closest existing objects increases;
    - `ioout` does not penetrate the boundary;

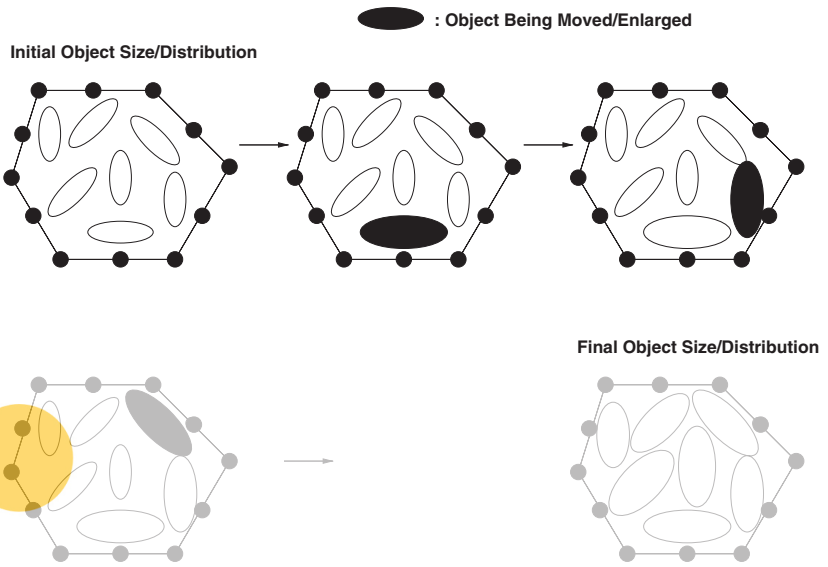

Figure 5. Movement and enlargement of objects.


```
  ◦ Enlarge object ioout by a small percentage
    - If the enlarged object ionew penetrates other objects or the boundary: revert to
      original size;
      ⇒ skip ioout;
enddo
```


The increase factors are progressively decreased for each new loop over the objects. Typical initial increase ratios are 5%. As the movement of objects is moderate (e.g. less than the radius for spherical objects), the spatial search data structures (bins, octrees) required during space filling can be reused without modification. We have found this move/enlarge post-processing to be very fast and effective, and use it routinely for the generation of DEM/DPM datasets.

6. ARBITRARY OBJECTS

The most time-consuming part of the present technique is given by the penetration/closeness checks. These checks are particularly simple for spheres. However, for arbitrary objects, the CPU burden can be significant. Specialized penetration/closeness checks are available for ellipsoids [15], but for general polyhedra the faces have to be triangulated and detailed face/face checks are unavoidable. The recourse taken here is to approximate arbitrary objects by a collection of spheres (see Figure 6). When adding a new object in space, the penetration/closeness checks are carried out for all spheres comprising the object. The new object is only added if all spheres pass the required tests.

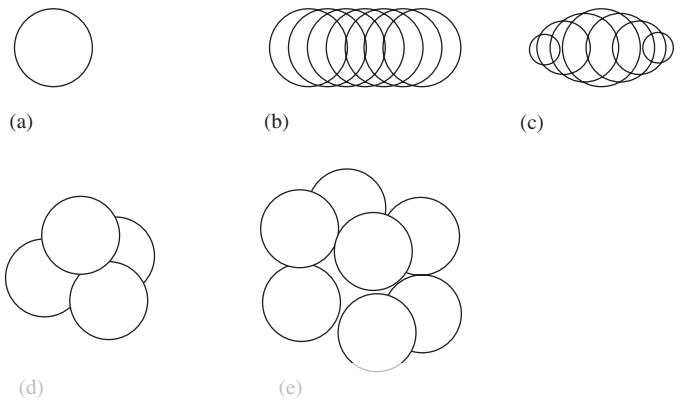

Figure 6. Arbitrary objects as a collection of spheres: (a) sphere; (b) tube; (c) ellipsoid; (d) tetrahedron; and (e) cube, etc.

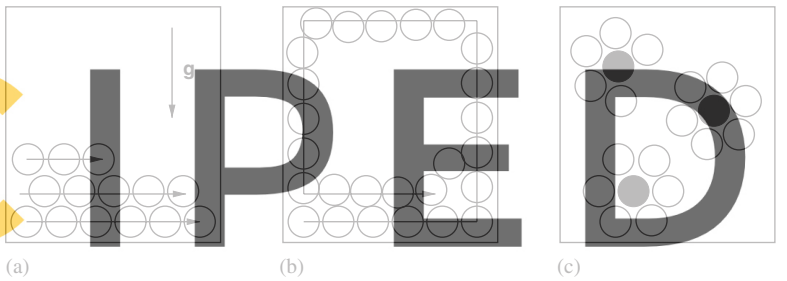

Figure 7. Deposition patterns: (a) fluid like; (b) layered growth; and (c) seed-point (radius) growing.

7. DEPOSITION PATTERNS

Depending on how the objects are removed from the active front, different deposition patterns can be achieved. The present technique always removes the object with the smallest average distance (size) to new neighbours from the active front. Different size distributions will therefore lead to different deposition patterns (see Figure 7). Gravitational deposition can be achieved by specifying a size distribution that decreases slightly in the direction of the gravity vector. The objects that are at the 'bottom' will then be removed first from the active front and surrounded by new objects. The same technique can be applied if magnetic fields are present. Layered growth can be obtained be assigning a slight increase in size based on the object number:

$$\delta = (1 + \varepsilon n_{\text{obj}})\delta_0 \tag{1}$$

where $\delta, \delta_0, n_{\text{obj}}, \varepsilon$ denote the size used, original size, object number and a very small number (e.g. epsilon $= 10^{-10}$). So-called radius growing, used to simulate granules or stone [13], can be achieved by first generating a coarse cloud of objects, and then using layered growth around each of these.

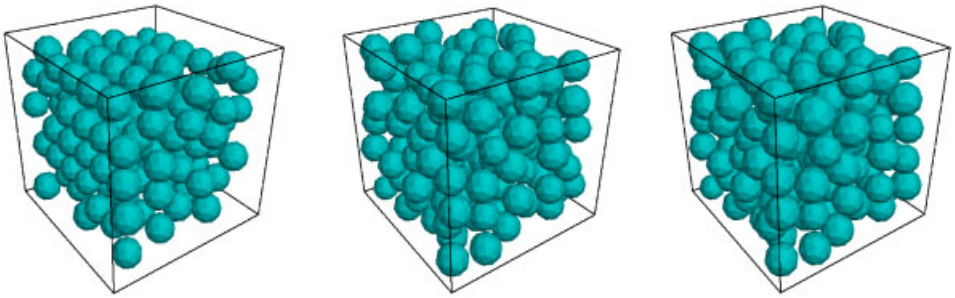

Figure 8. Cube filled with spheres.

8. EXAMPLES AND TIMINGS

The proposed advancing front object generation algorithm has been used to generate object clouds for many applications: FPM/SPH simulations for continua, DEM/DPM simulations for discontinua, granular media, etc. Of these, we show a representative cross-section. All the object clouds were generated on a PC with Intel P4 chip running at 3.2 Ghz, 1 GByte Ram, Linux OS and Intel Compiler.

8.1. Cube

This first configuration considered is a cube of size $1 \times 1 \times 1$, and is used to highlight the effectiveness of the optimal packing procedures described, and to obtain timing trends. The prescribed sphere size was $\delta = 0.15 \pm 0.015$, and a uniformly random variation of the size was allowed. Figures 8(a)–(c) show the sphere distributions obtained with simple stencil placement, stencil placement with closest object movement, and stencil placement with closest object movement and move/enlarge post-processing. The number of spheres generated were $n_s = 133, 169, 169$, and the volume fractions $v = 0.291, 0.410, 0.448$. As one can see, the differences are significant. Ultimate volume fill may not be important for FPM/SPH applications, but plays an important role for DEM simulations [13].

The generation times, on the other hand, vary considerably depending on the amount of effort spent for optimum packing. The generation of 100K spheres in the unit cube requires 33 s for simple stencil placement and 120 s for stencil placement with closest object movement and move/enlarge post-processing, i.e. almost four times as much. By contrast, the generation of a mesh with 100K points (and 560K elements) requires approximately 100 s, i.e. considerably more than the generation of spheres via simple stencil placement, but comparable to the time required for stencil placement with closest object movement and move/enlarge post-processing. It appears that the extra cost involved in closest object movement and move/enlarge post-processing is comparable to the volume coherency (elements crossing front) checks required for grid generation.

8.2. F117

A cloud of points for the aerodynamic simulation of inviscid, transonic flow past (half) an F117 fighter via FPM [9] is considered next. The point density was specified through the combination of a background grid and background sources. The surface triangulation consisted

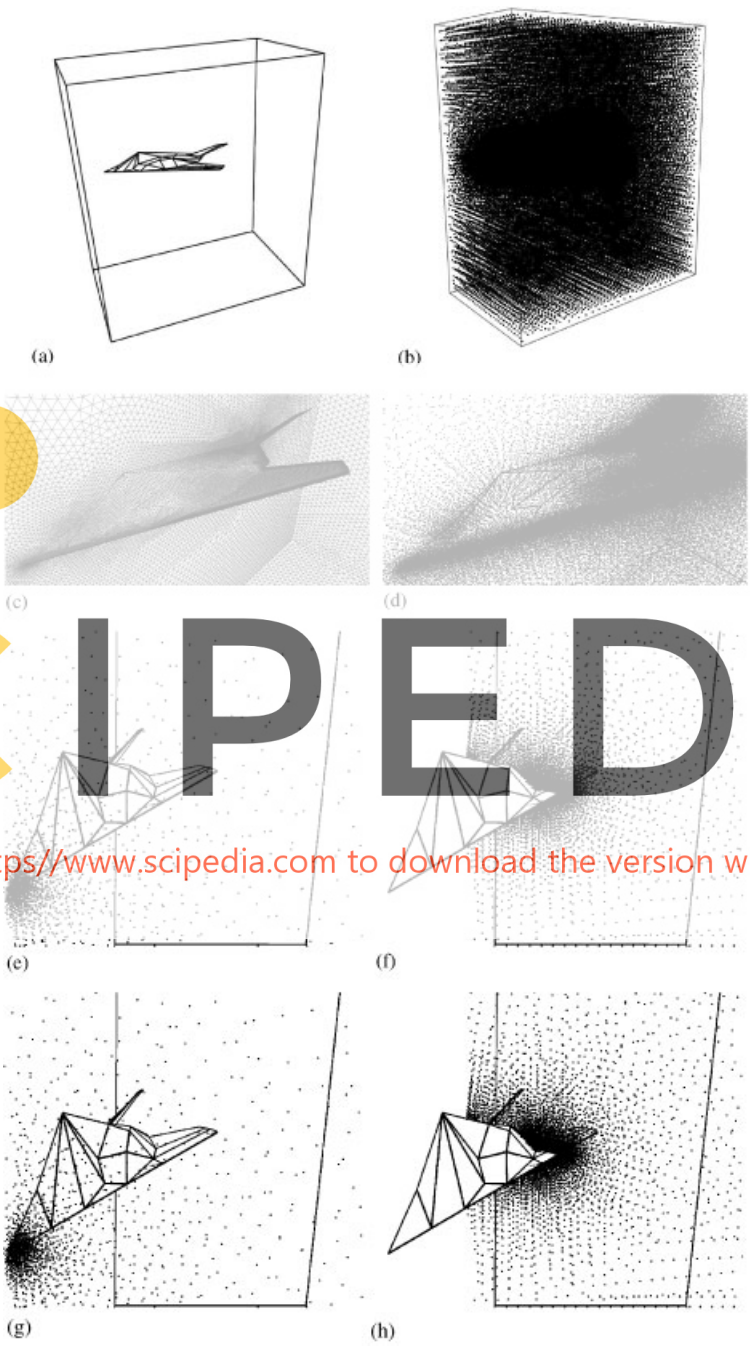

Figure 9. (a,b) F117: CAD definition and global cloud of points. (c,d) F117: close-up of surface mesh and cloud of points; (e,f) F117: cuts at $x = 0, 120$; and (g,h) F117: cut at $x = 120$ (detail) and $x = 190$.

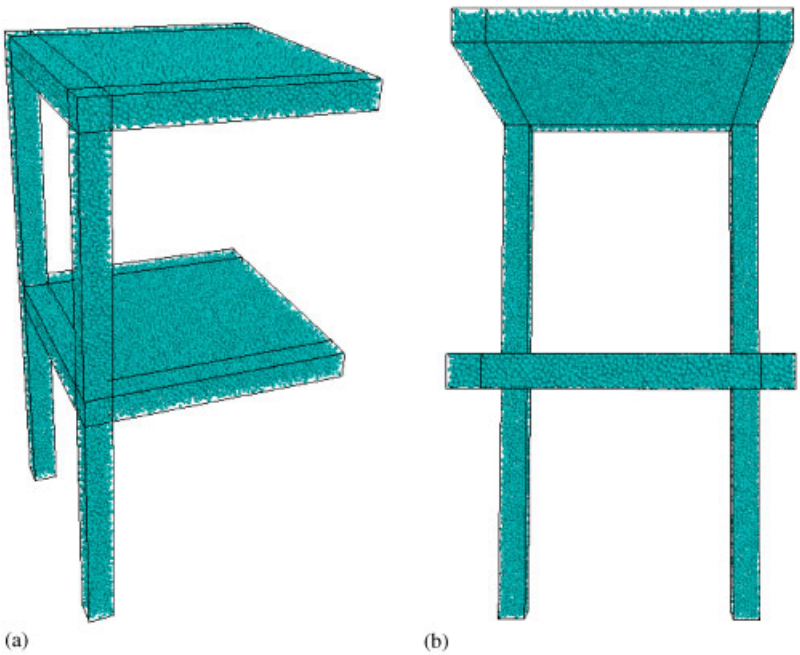

Figure 10. DEM discretization of civil engineering structure.

of approximately 50 Kpts and 100 Ktria. The advancing front point generator added another 200Kpts using simple stencil placement. Figure 9 shows the CAD definition of the computational domain, the global cloud of points, a close-up of the surface mesh, the cloud of points close to the plane, as well as some slices through the volume. The spatial variation of point density is clearly visible. The complete generation process (boundary triangulation, volume fill, output of files, etc.) took 44 s.

8.3. Civil engineering structure

Ongoing research in concrete modelling via discrete element methods requires the discretization of realistic structures for validation and verification. Figure 10 shows a typical civil engineering structure, where columns and floors are present. The structure was discretized with spheres that varied in diameter $\pm 20\%$ using stencil placement with closest object movement and move/enlarge post-processing. This resulted in approximately 41 883 spheres, a volume fill ratio was $v = 0.495$, and an average number of contacts $n_c = 5.71$. The generation time was 127 s.

8.4. Hopper with beans/ellipsoids

Granular materials that require simulation include grains, ground stone, woodchips, and many other materials [10, 12]. Bridging in silos and hoppers can cause severe structural damage, and has always been a concern. Figure 11 shows a hopper configuration with bean-like objects composed of four spheres each. The bean-like objects are obtained by placing spheres at the

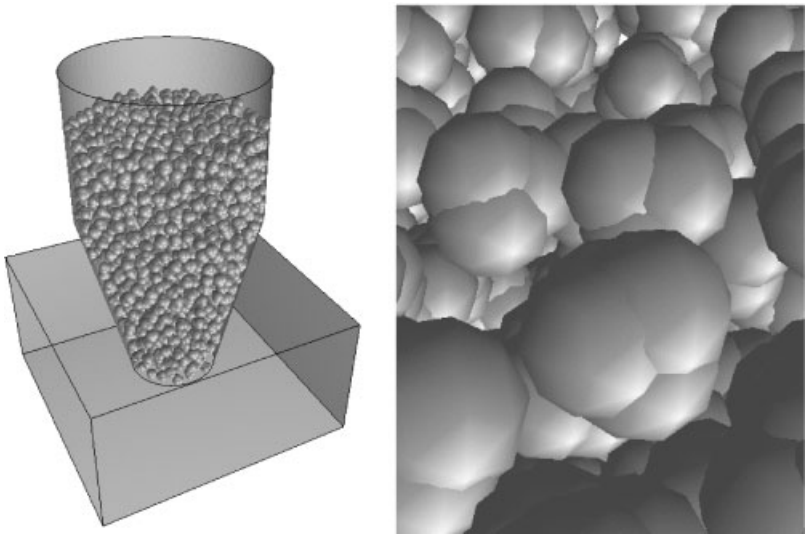

Figure 11. Hopper filled with beans.

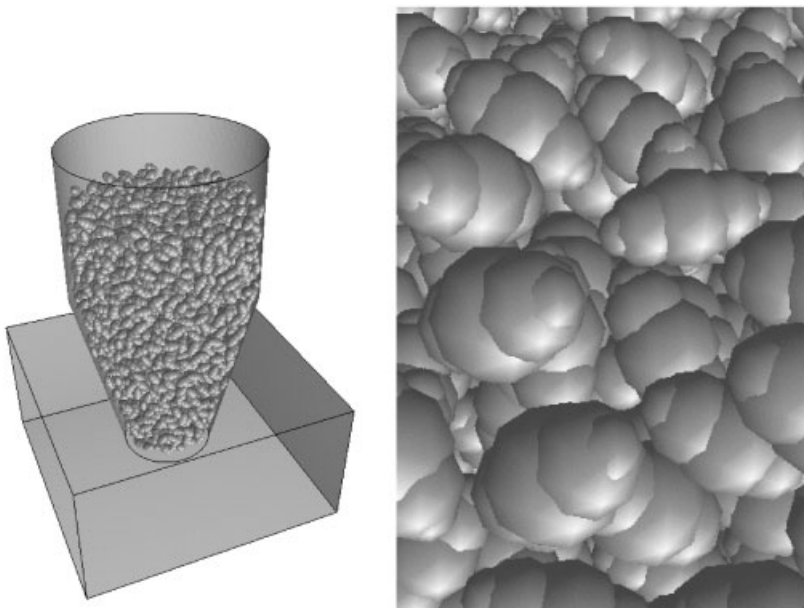

Figure 12. Hopper filled with ellipsoids.

points of perfect tetrahedra, and then enlarging them in a suitable way to obtain the desired shape. The total number of beans is 2124, i.e. 8496 spheres, and took 7 s to generate. Figure 12 shows the same configuration filled with ellipsoidal objects composed of five spheres each. The

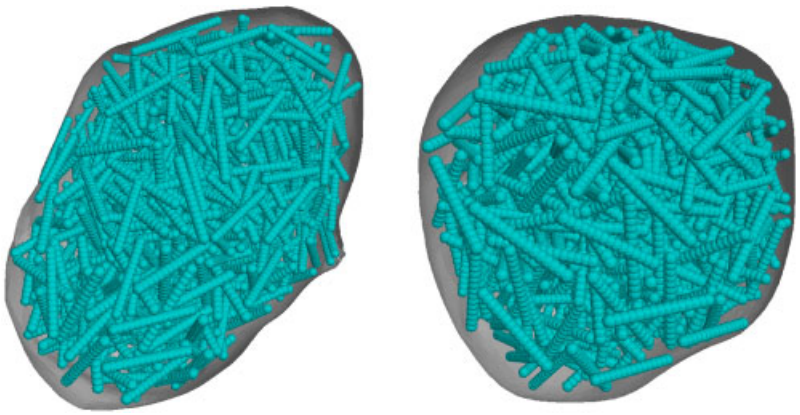

Figure 13. Aneurysm filled with coils.

objects are obtained by placing five spheres along a line, and then enlarging them appropriately to obtain the desired shape. The total number of ellipsoids is 2794, i.e. 13970 spheres, and took 10 s to generate.

8.5. Aneurysm with coils

A modern interventional procedure for aneurysms is coiling. In order to estimate the effectiveness of different volume-fill ratios, the flowfield of an aneurysm was filled with different volume-densities of coils and computed. Figure 13 shows the aneurysm, which is approximately $2 \times 2 \times 1.5$ cm in size, and (for visualization) a configuration of coils with twice the size as used in the simulations. The coils were simulated as tubes consisting of 15 spheres. The configuration shown consists of 750 coils, 11 250 spheres, has a volume fill ratio of approximately $v = 0.15$, and took 8 s to generate.

9. CONCLUSIONS AND OUTLOOK

A general advancing front technique to fill volumes with arbitrary objects has been developed. The input required consists of the specification of the desired mean distance between objects in space and an initial triangulation of the surface. One object at a time is removed and, if possible, surrounded by admissible new objects. This operation is repeated until no active objects are left. The scheme, when used as a point generator for meshless solvers (SPH, FPM) is considerably faster than volume grid generators based on the advancing front technique. The main reason for this disparity in speeds is the extra volume coherency (elements crossing front) check, which is costly. These checks are not required for clouds of points. This observation confirms, to a certain degree, the premise that point generation is simpler than element and point generation.

Two techniques to obtain maximum packing are discussed: closest object placement (during generation) and move/enlarge (after generation). The first one of these nearly triples the

required CPU time, but is still very competitive in comparison to direct contact calculations and deposition via gravity.

It was also shown how different deposition or layering patterns can be achieved by selecting the order in which objects are eliminated from the active front.

Although very competitive, the present procedure may be parallelized along the lines of unstructured grid generators with the advancing front technique [27], i.e. via domain decomposition.

Future work will consider the generation of clouds of points that exhibit a high degree of spatial anisotropy, such as those required for Reynolds-averaged Navier–Stokes simulations, the fast selection of local clouds of points allowing for different least-squares based polynomical interpolations of the point unknowns, and the extension to other classes of general objects.

ACKNOWLEDGEMENTS

A considerable part of this work was carried out while the first author was visiting the Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona, Spain, in the Summer of 2003. The support for this visit is gratefully acknowledged.

REFERENCES

1. Nay RA, Utku S. An alternative for the finite element method. *Variational Methods in Engineering* 1972; **1**.
2. Batina J. A gridless Euler/Navier–Stokes solution algorithm for complex aircraft configurations. *AIAA-93-0333*, 1993.
3. Belytschko T, Lu Y, Gu L. Element free Galerkin methods. *International Journal for Numerical Methods in Engineering* 1994; **37**:229–256.
4. Duarte CA, Oden JT. H_p clouds—a meshless method to solve boundary-value problems. *TICAM-Report 95-05*, 1995.
5. Gingold RA, Monahghan JJ. Shock simulation by the particle method SPH. *Journal of Computational Physics* 1983; **52**:374–389.
6. Oñate E, Idelsohn S, Zienkiewicz OC, Taylor RL. A finite point method in computational mechanics. Applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering* 1996; **39**:3839–3866.
7. Oñate E, Idelsohn S, Zienkiewicz OC, Taylor RL, Sacco C. A stabilized finite point method for analysis of fluid mechanics problems. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:315–346.
8. Liu WK, Chen Y, Jun S, Chen JS, Belytschko T, Pan C, Uras RA, Chang CT. Overview and applications of the reproducing kernel particle methods. *Archives of Computational Methods in Engineering* 1996; **3**(1): 3–80.
9. Löhner R, Sacco C, Oñate E, Idelsohn S. A finite point method for compressible flow. *International Journal for Numerical Methods in Engineering* 2002; **53**:1765–1779.
10. Cundall PA, Stack ODL. A discrete numerical model for granular assemblies. *Geotechnique* 1979; **29**(1): 47–65.
11. Cundall PA. Formulation of a three-dimensional distinct element model—part I: a scheme to detect and represent contacts in a system composed of many polyhedral blocks. *International Journal of Rock Mechanics and Mining Sciences* 1988; **25**:107–116.
12. Cleary PW. Discrete element modeling of industrial granular flow applications. *TASK. Quarterly Scientific Bulletin* 1998; **2**:385–416.
13. Sakaguchi H, Murakami A. Initial packing in discrete element modeling. In *Discrete Element Methods*, Cook BK, Jensen RP (eds). ASCE: New York, 2002; 104–106.
14. Löhner R, Oñate E. An advancing point grid generation technique. *Communications in Numerical Methods in Engineering* 1998; **14**:1097–1108.
15. Feng YT, Han K, Owen DRJ. An advancing front packing of polygons, ellipses and spheres. In *Discrete Element Methods*, Cook BK, Jensen RP (eds). ASCE: New York, 2002; 19–98.

16. Yerry MA, Shepard MS. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering* 1984; **20**:1965–1990.
17. Löhner R, Parikh P. Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Fluids* 1988; **8**:1135–1149.
18. Peraire J, Morgan K, Peiro J. Unstructured finite element mesh generation and adaptive procedures for CFD. *AGARD-CP-464*, 18, 1990.
19. George PL, Hecht F, Saltel E. Fully automatic mesh generation for 3D domains of any shape. *Impact of Computing in Science and Engineering* 1990; **2**(3):187–218.
20. Shepard MS, Georges MK. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering* 1991; **32**:709–749.
21. George PL, Hecht F, Saltel E. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering* 1991; **92**:269–288.
22. Weatherill NP. Delaunay triangulation in computational fluid dynamics. *Computational Mathematics and Applications* 1992; **24**(5/6):129–150.
23. Weatherill N, Hassan O, Marchant M, Marcum D. Adaptive inviscid flow solutions for aerospace geometries on efficiently generated unstructured tetrahedral meshes. *AIAA-93-3390-CP*, 1993.
24. Löhner R. Extensions and improvements of the advancing front grid generation technique. *Communications in Numerical Methods in Engineering* 1996; **12**:683–702.
25. Löhner R. Automatic unstructured grid generators. *Finite Elements in Analysis and Design* 1997; **25**:111–134.
26. George PL, Borouchaki H. *Delaunay Triangulation and Meshing*. Hermes: Paris, 1998.
27. Löhner R. A parallel advancing front grid generation scheme. *International Journal for Numerical Methods in Engineering* 2001; **51**:663–678.

